

Terminology mining with ATA* and Galinha

Joana L. Paulo, David M. de Matos, and Nuno J. Mamede

L²F – Spoken Language System Laboratory
INESC-ID Lisboa/IST, Rua Alves Redol 9, 1000-029 Lisboa, Portugal
<http://www.l2f.inesc-id.pt/>
{joana.paulo,david.matos,nuno.mamede}@l2f.inesc-id.pt

Abstract. In this article we describe how we joined ATA and Galinha to ease terminology extraction tasks. We open with a brief introduction to the field, then we present the terms extractor and the web interface. Finally we discuss their integration. Shortly, ATA is a system for automatic terms acquisition that takes a text from a specific field and analyzes it in order to decide which of the detected nouns and noun phrases ought to be considered terminological units. Currently, ATA is being evaluated over a Portuguese nautical corpus. Galinha is a system that integrates multiple linguistic resources and tools. Galinha enables easy module integration and testing of prototypical configurations, thereby reducing the effort and backtracking usual in the construction of modular applications. Joining ATA and Galinha allowed us to provide a graphical web interface to make it easier to automatically acquire terms while enabling access to the intermediate results of each module.

1 Introduction

First of all, let us clarify what do we refer to as a term; present the main techniques described in the literature and list the evaluation methods for automatic terms acquisition.

1.1 Meaning of term

Since the word *term* is ambiguous, we shall make clear from the beginning that in this context, we will consider it as a linguistic representation of some concept by means of a simple noun or a noun phrase [11]. We consider two term types: simple and compound. Other phraseological structures characterizing some knowledge domains are not in the scope of our system.

Simple terms consist of a single lexical unit: a graphical word. The complexity associated with the detection of this kind of terms arises from their unremarkable appearance. This means that there is no way for one to be distinguished from another, unless the system has an analyzer of morphological structure which can sort term-candidates by the occurrence of specific traits. This is not our case.

Compound terms consist of more than one lexical unit (graphical form). Thus, they are less prone to ambiguity than simple terms. Nevertheless, they require a previous syntactical study to verify whether a set of words actually defines a term's syntactical structure.

* ATA has been partially supported by the Fundação para a Ciência e Tecnologia under project number PLUS/1999/LIN/15150

1.2 Using static information

All lexical units have an associated frequency, corresponding to the number of times they appear in a corpus, that can be easily measured. According to several works [15, 12, 3, 11], using this information, we can decide whether a word is a term: items that are nouns and that appear more than a given number of times may be considered as candidates to simple terms; words with other categories must be kept in order to complete the processing of compound terms. One of ATA's goals is to simplify this task, working with both types of terms with the same tools and in the same way, merging the two processes.

1.3 Main techniques

Most systems designed for this kind of task take a plain text and extract from it a list of candidate terms. To make the terminologist's task easier, this list is provided with its context and assorted additional information (such as relative frequency for that word and for its root). There are three main techniques used for this problem: the statistical, the linguistic and the hybrid.

Statistical systems detect lexical units whose frequency is higher than a given corpus-based threshold definition. The problem with this approach is that it fails to detect low-frequency terms.

Linguistic systems detect recurrent patterns from complex terminological units such as noun-adjective and noun-preposition-noun. Patterns to be detected are assumed to have been designed by linguists.

Hybrid systems start by detecting some basic linguistic structures, such as noun or prepositional phrases, and then, after the candidate terms have been identified, the relevant statistical information is used to decide whether they correspond to terms. This will be our methodology.

1.4 Evaluation

The development of noun phrase extractors is a very delicate task constrained by robustness and accuracy.

Robustness is subject to a strong restriction: it should be possible to be used over a wide range of unrestricted texts gathered from large corpora. This means that it has to be domain-independent, that is, it cannot use any a priori semantic or conceptual information. Syntactically the task is harder when generic as we do not take in account the specific restricted surface structures. There are cases in the literature [9] where the use of restricted syntactical structures improved the results when extracting medical terms. The idea is to reduce the possible nominal structures that are allowed to be terms according to a restricted field.

Accuracy is also an issue because the noun phrases extracted by the system are the candidate terms that will be proposed to the user building a domain's terminology.

The two most frequently used metrics in the evaluation of this type of system are precision and recall [13]. Recall is defined as the relationship between the sum of retrieved terms and the sum of existing terms in the document that is being explored. Precision accounts for the relationship between extracted candidate terms that are actually terms and the aggregate of candidate terms that are found. These metrics may be seen as the capacity to extract all terms from a document (recall) and the capacity to discriminate between those units detected by the system which are terms and those which are not (precision).

2 Terms acquisition with ATA

ATA [17, 16] is a system for Automatic Term Acquisition, that processes technical texts and produces a list of noun phrases likely to be terminological units. ATA's architecture is based on the well-known architecture proposed by Daille [5]. ATA consists of three main modules (see figure 1): (i) linguistic enrichment and selection of those units that may be terms due to their syntactical categories; (ii) enrichment of candidates with corpora-based statistical information; and (iii) decision about whether they are terms and should be proposed to the user.

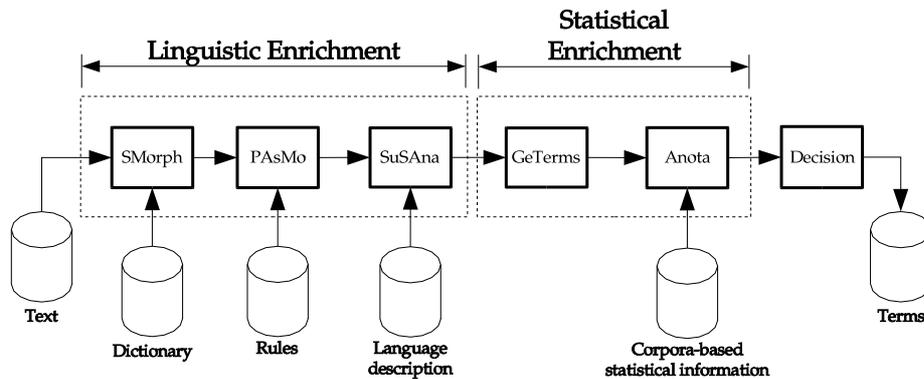


Fig. 1. ATA's architecture.

2.1 Linguistic analysis sequence

In the linguistic analysis sequence, we use three tools¹ to lemmatize and morphologically annotate the text using a dictionary; to rewrite it according to some rules; and to group phrase constituents. A brief description of each one follows.

`SMorph` [1] enriches each word with its morphological characterization according with a dictionary: a collection of words with their morphological characteristics and their linguistic roots. Since `SMorph` allows the construction of large dictionaries required by the linguistic analysis of texts, and since our linguistic data were poor, we had to improve the dictionaries in order to enhance the results.

`PASMo` [18] rewrites the text according to recomposition and correspondence rules based on the morphological features of the words. `PASMo` also breaks the text into sentences (according to selected punctuation marks). `PASMo` is responsible for the rewriting of dates, compound nouns, consecutive unknown words, numbers, and so on, minimizing ambiguity. `PASMo` may also be used to translate tags, thus facilitating the connection with the syntactic analyzer. We had to rebuild an old system [10] from scratch in order to improve its efficiency and expressiveness.

`SuSAna` [2] is a surface syntactic analyzer that groups the words of each phrase in the text into syntactic chunks. `SuSAna`'s chunks are described in a surface grammar. `SuSAna` not only gives us the usual syntactic chunks, but also provides the means to describe a hierarchy of categories. Using an hierarchical description, we simplified the problem and managed to treat both types of terms in the same process: we declared a term as being a simple or a compound term; and a simple term as being a noun; and a compound term as a noun phrase. This approach enabled us to reuse the syntactic descriptions available for written Portuguese. `SuSAna` also gives us all the possible structures detected on the phrase, even if they overlap (or, if needed, the biggest structure found in each phrase, according to the syntactic description).

2.2 Statistical enrichment sequence

`GeTerms` is a tool that selects those structures with a syntactical structure that could be terms. These are, for Portuguese, all noun phrases present on the text. Besides selecting term candidates, `GeTerms` counts their occurrences, providing information to the next stage that compares the occurrences on the specialized text with those from the "normal" one. Since `SuSAna` did most of the linguistic filtering job by choosing what we defined as terms, we only need to select the relevant information and count the occurrences relative to the given text.

`Anota` is a merging tool. This tool enriches each candidate term with corpora-based statistical information. This information refers to the occurrence of the word and root considered a candidate term.

Note that this task is strongly constrained by memory usage. This means that a way had to be found to avoid reading all the data into memory. Otherwise it would have

¹ A fourth tool can be used (between `SMorph` and `PASMo`) to help solving the morphological ambiguity according to statistic information. We tried to use `MARV` [21] for this ambiguity resolution without success (further details can be found in [20]).

been impossible to use this tool, especially when creating the corpora-based statistical information from the amounts of data in question (one month of newspaper text has about 1.3 million nouns and noun phrases).

2.3 Decision stage

`Decision` is a module that evaluates the candidate lists, producing the final results to be presented to the user. `Decision` compares the occurrence of candidate terms in the specialized text and their occurrence on newspaper corpora analyzed by the same chain process. The corpora-based occurrence information is available after the linguistic and statistical chains. `Decision` was made independent to ease the study of the best parameters to use when comparing both values of occurrences.

The output of `Decision` is a list of words that may be terms. This list may be divided into two sets, both of which may be empty: the first set contains simple term candidates, identified in the text; the second set contains compound term candidates.

Even though the two types of terms to be detected (simple and compound) have different characteristics, we handle them in the same way, by delegating on the grammar the responsibility for customized processing. In an hybrid system such as this, high-frequency terms will be detected statistically, while low-frequency terms will be detected through the terms grammar. Afterwards, it will be necessary to review the candidate terms. This revision is always necessary since not even human annotators always agree on the terms in a text.

2.4 Evaluation

For evaluation purposes, we analyzed a 114,000-word corpus and asked the system to detect their terms. Then we compared the computed list with a list of terms manually detected by linguists.

We are still running tests and we are trying to experimentally find the best parameters according to which we will say that a noun phrase or noun is a term: the minimal number of occurrences that a term should have and the multiplicative factor when comparing the occurrence on corpora to the occurrence on the specialized text (a candidate has to appear more than n times in the specialized text than in the corpora to be considered as a term by the system).

Since ATA is still under development, we can not present the concluding results yet. However, we already know that we have to improve the linguistic data used on the first linguistic enrichment in order to reduce the noise level.

We also found that some terms are not detected because they appear (against our base theory) only once. That opened a path to study the possible use of ontologies. The two fields will gain from the merging: terms detection can help building new conceptual descriptions of some specialized field using information extracted from related literature; concept descriptions can fill the gap when the detection of a term by enforcing the importance of their relative terms.

3 Web-based interface with Galinha

Galinha [7] (Galaxy Interface Handler) is a web-based interface that simplifies access to modules, applications, and library interfaces: it enables users to access and compose modules using a web browser.

The infrastructure used to support the interface is a partial implementation² of the theoretical interconnection model presented in [6]. The Galaxy Communicator system [14] was selected to provide messaging support for the infrastructure's message exchanges.

Galinha's Galaxy-based general architecture is shown on figure 2.

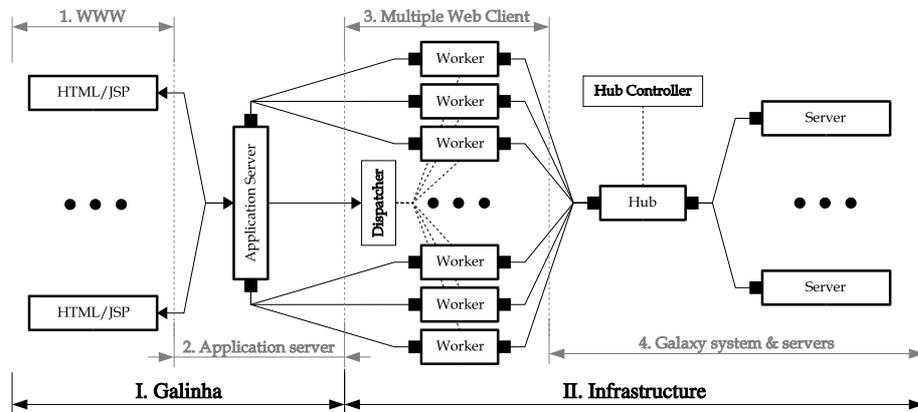


Fig. 2. The Galaxy infrastructure, control servers, and user-side levels.

The application server is one of the interface's key components. It provides the runtime environment for the execution of the various processes within the web application. Moreover, it maintains relevant data for each user, guarantees security levels and manages access control. The interface also uses the application server as a bridge between the interface's presentation layer (HTML [24]/JavaScript [8], at the browser level) and the infrastructure.

The presentation layer consists of a set of PHP scripts and classes [19] and related pages. It is built from information about the location (host and port) of the MultipleWebClient that provides the bridge with the Galaxy system the user wants to contact; and from XML [23] descriptions of the underlying Galaxy system (provided by the hub controller). This is a remake of a previous Java/servlets-based interface.

Besides allowing execution of services on user-specified data, the interface allows users to create, store, and load service chains. Service chains are user-side service- or program sequences provided by the servers connected to the infrastructure: each service

² Currently, the main capabilities have been implemented, but functionality like message type checking is as yet unfinished.

is invoked according to the user-specified sequence. Service chains provide a simple way for users to test sequences of module interactions without having to actually freeze those sequences or build an application. The interface allows not only inspection of the end results of a service chain, but also of its intermediate results. Service chains may be stored locally, as XML documents, and may be loaded at any time by the user. Even though, from the infrastructure's point of view, service chains simply do not exist, selected service chains may be frozen into system-side programs and become available for general use. Other developments on the user side are currently being studied to address sharing of chain configurations without infrastructure support.

3.1 Using the web interface

To use the interface, users must first specify the location – host and port – of the back-end system providing the services. Then, the interface presents the main view, divided into four areas (see figure 3): the top one provides a general control menu; this frame is always available. The left frame presents descriptions of back-end systems and allows access to servers (each of which in turn has additional subdivisions) and programs at any time. The frame to the right presents the user's services chains, if any. The main frame (also the main input area) presents various states of the system or of its interaction with the user.

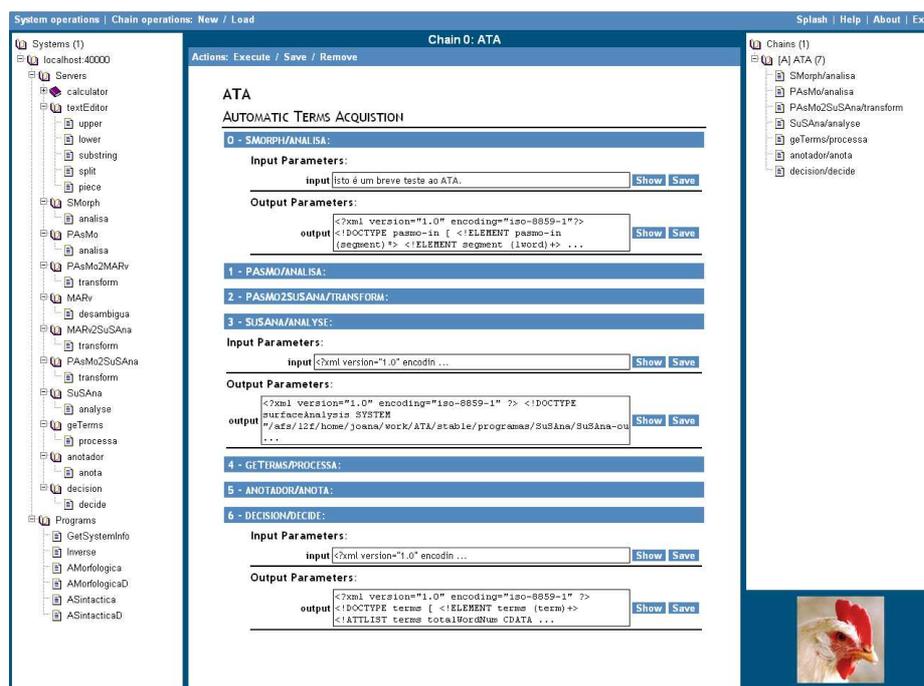


Fig. 3. Galinha as graphical interface for ATA.

When a service is selected, its description is presented to the user, stating the input and output ports, as well as a description of its actions. Service selection also presents the user with the list of possible operations on that module.

On the right hand frame, service chains are shown, along with the list of modules in each one, as well as the state of their interconnections.

Each service in a chain can be in one of two states: complete, i.e., all input and output connections have been specified; or incomplete (the default). This is made apparent to the user during the interaction with a chain. A chain becomes executable when all of its services are complete. After execution of a service chain, resulting data (both partial and final) may be viewed in the web browser or saved to a file. This is an important feature when testing a system as it avoids unnecessary server/infrastructure calls and improves bandwidth usage.

3.2 Module definition

Modules may be included in the infrastructure in two ways: the first is to create the module anew or to adapt it so that it can be incorporated into the system; the second is to create a capsule for the existing module – this capsule then behaves as a normal Galaxy server would.

Whenever possible or practical, we chose the second path. Favoring the second option proved a wise choice, since almost no changes to existing programs were required. In truth, a few changes, mainly regarding input/output methods, had to be made, but these are much simpler than rebuilding a module from scratch: these changes were caused by the requirement that some of the modules accept/produce XML data in order to simplify the task of writing translations. This is not a negative aspect, since the use of XML as intermediate data representation language also acts as a normalization measure: it actually makes it easier for future users to understand modules' inputs and outputs. It also eases the eventual conversion between module's outputs and inputs (also needed on a command-line approach).

4 ATA and Galinha joined to ease terms acquisition

We had a chain of modules without any interface and a web-based interface with multiple-servers support to test. Most of the modules initially included in Galinha, i.e., before ATA, were simply for test purposes and were not, in any case, very complex. The first practical test took place when production modules had to be incorporated into the system. Several modules, namely those needed by ATA were added. Writing general adapter modules (such as data format converters) was also required. In both cases, the work to be done proved to be simple (only data format manipulations were required).

Since Galinha works with chains, to include modules for supporting ATA we had to write the corresponding chain and to connect the modules. The main linguistic analysis modules used by ATA were made available through Galinha individually so that they can be used later for other purposes. As the programs were client-server applications we extended the client of each application to be a Galaxy server. Also, a wrapper was written to call external applications. The wrapper was so simple we were

able to generalize it to use any future application we may need. To actually connect the modules we had only to connect them as a chain, making use of the interface. This proved to be a simple task.

Figure 3, shows *Galinha* with ATA's definition: on the left, we have the available systems; on the right, we have a chain where all the relevant services are connected and is ready for execution; in the middle frame, we can give the text that we want to analyze and – after the results are produced – browse each module's input and output. Since the final result depends on the intermediate results, their availability makes evaluation easier.

We found that a client-side cache mechanism would improve the efficiency when changing some values in the middle of the chain: the chain would process from the first difference made instead of (re)processing all the data thus improving the response time.

5 Conclusions

Our objective is to automatically extract terms. Also, we wanted to do it with some graphical interface that would focus exclusively on the data flowing to/from of each module, without regard for module internals, including the implementation language or internal data representation. That would allow us to provide the service without the requirement for expertise in distributed systems. Note that the whole ATA system was built by someone who had no previous experience with the infrastructure. The learning curve was both fast and comfortable (the whole integration task including building the adaptors took approximately three days).

The ATA system can be a useful helper solving the problem of the semi-automatic terms acquisition need for important task such as the building of terminological indexes, translations or text categorization. Besides, it provides a base framework to work over written corpora since it joins the main modules needed for such task.

ATA will be evaluated using the described methods (see §2.4). We hope to achieve results similar to those of systems for Portuguese [4, 22] and for other foreign languages as English [12] and French [5].

After designing ATA and its modules, we found that *Galinha* could do the modules integration job providing a graphical interface easy to use, where adding new modules is simple. Besides that, *Galinha* gives us access to intermediate results and can be made available to anyone on the web what reinforce the choice.

Galinha proved to be a useful interface for application development, since their modules can be almost anything and run almost anywhere, as long as a communication channel can be established between them. Also, the use of text frames as a communication media allows for flexible module deployment. As described, the infrastructure is now richer, since the new modules become available for use in other contexts.

The interface, by hiding most of the complexities of the underlying system and of the modules attached to it, empowers non-expert users to play with various scenarios and to investigate possible differences. This is particularly important in environments such as schools, in which students have to become acquainted with the tools relevant for a particular field; and whenever it is desirable to have a fast learning curve, e.g. when new people integrate a project team.

The underlying model is useful not only in helping in application construction, but also as a guide to thinking about modular applications: the interface materializes this aspect, since it allows non-expert users to successfully design applications and application components.

Regarding future developments in the interface: while the one described was aimed only at module users, another is under development. The new interface is aimed at helping module developers integrate their work for use in the system.

Acknowledgements

We would like to acknowledge the work by João Graça and Alexandre Mateus on the web interface and the work by Ricardo Ribeiro and Fernando Batista on the linguistic tools.

References

1. Salah Aït-Mokhtar. *L'analyse Présyntaxique en une seule étape*. PhD thesis, Université Blaise Pascal, Clermont-Ferrand, GRIL, 1998.
2. Fernando Batista and Nuno Mamede. Flexible module for shallow parsing, using preferences. In António Branco, Amália Mendes, and Ricardo Ribeiro, editors, *Tagging and Shallow Processing of Portuguese: workshop notes of TASHA'2003*, volume TR-03-28 of *Technical Reports*, pages 5-6, Lisboa, Portugal, 2003. Faculdade de Ciências da Universidade de Lisboa.
3. D. Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases. *Proceedings of the 15th International Conference on Computational Linguistics, COLING'92*, 1992. p. 977-981.
4. J. Ferreira da Silva and G. Pereira Lopes. A local maxima method and a fair dispersion normalization for extracting multi-words units from corpora. *International Conference on Mathematics of Language, Orlando, July 1999*.
5. B. Daille. Study and implementation of combined techniques for automatic extraction of terminology. *The balancing act combining symbolic and statistical approaches to language*, pages 49-66, 1996.
6. David M. de Matos, Alexandre Mateus, João Graça, and Nuno J. Mamede. Empowering the user: a data-oriented application-building framework. In *Adj. Proc. of the 7th ERCIM Workshop "User Interfaces for All"*, pages 37-44, Chantilly, France, October 2002. European Research Consortium for Informatics and Mathematics.
7. David M. de Matos, Joana L. Paulo, and Nuno J. Mamede. Managing Linguistic Resources and Tools. In N. J. Mamede, J. Baptista, I. Trancoso, and M. das Gracas Volpe Nunes, editors, *Computational Processing of the Portuguese Language - Proc. of the 6th Intl. Workshop, PROPOR 2003*, number 2721 in *Lecture Notes in Artificial Intelligence*, pages 135-142, Faro, Portugal, June 26-27 2003. Springer-Verlag, Heidelberg. ISBN 3-540-40436-8.
8. ECMA International, Geneva, Switzerland. *Standard ECMA-262 - ECMAScript Language Specification*, 3rd edition, December 1999. See also: www.ecma.ch.
9. Rosa Estopà. *Les unitats terminològiques polilexemàtiques en els lèxics especialitzats: dret i medicina*. PhD thesis, Institut Universitari de Lingüística Aplicada, Barcelona, UPF, 1999.
10. Abbaci Faiza. Développement du module post-smorph. Master's thesis, Mémoire de DEA de linguistique et informatique, GRIL, Université Blaise Pascal, Clermont-Ferrand, 1999.

11. C. Jacquemin and D. Bourigault. Term extraction and automatic indexing. *R. Mitkov, editor, Handbook of Computational Linguistics*, 2000.
12. J. S. Justeson and S. M. Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, p. 9-27, 1995.
13. C. D. Manning and H. Shutze. *Foundations of Statistical Natural Language Processing*. MIT Press, London, 1999.
14. Massachusetts Institute of Technology (MIT), The MITRE Corporation. *Galaxy Communicator (DARPA Communicator)*. See: <http://communicator.sf.net>.
15. A. P. Marquez Neto. Terminologia e corpus linguístico. *Revista Internacional de Língua Portuguesa - RILP n. 15*, p. 100-108, 1996.
16. Joana L. Paulo, Margarita Correia, Nuno J. Mamede, and Caroline Hagège. Using morphological, syntactical, and statistical information for automatic term acquisition. In Elizabete Ranchhod and Nuno Mamede, editors, *Advances in Natural Language Processing, Third International Conference, Portugal for Natural Language Processing (PorTAL)*, pages 219–227, Faro, Portugal, 2002. Springer-Verlag, LNAI 2389.
17. Joana L. Paulo and Nuno J. Mamede. ATA - Automatic Term Acquisition. In *Proceedings of the Workshop on Extraction of Knowledge from Databases*, pages 51–54, Porto, Portugal, 2001.
18. Joana Lúcio Paulo. PAsMo – Pós-Análise Morfológica. Relatório técnico, Instituto Superior Técnico, Lisboa, 2001.
19. PHP Group. *PHP Hypertext Processor*. See: www.php.net.
20. Ricardo Ribeiro, Nuno Mamede, and Isabel Trancoso. Reusing linguistic resources: a case study in morphosyntactic tagging. In António Branco, Amália Mendes, and Ricardo Ribeiro, editors, *Tagging and Shallow Processing of Portuguese: workshop notes of TASHA'2003*, volume TR-03-28 of *Technical Reports*, pages 31–32, Lisboa, Portugal, 2003. Faculdade de Ciências da Universidade de Lisboa.
21. Ricardo Ribeiro, Luís Oliveira, and Isabel Trancoso. Morphosyntactic Disambiguation for TTS Systems. In Manuel González Rodríguez and Carmen Paz Suárez Araujo, editors, *Proc. of the 3rd Intl. Conf. on Language Resources and Evaluation*, volume V, pages 1427–1431, Las Palmas, 2002. ELRA. ISBN 2951740808.
22. J. Silva, G. Dias, S. Guilloré, and G. Lopes. Using localmaxs algorithm for the extraction of contiguous and non-contiguous multiword lexical units. *9th Portuguese Conference on Artificial Intelligence*, 1695:113–132, September 1999.
23. World Wide Web Consortium (W3C). *Extensible Markup Language (XML)*. See: www.w3.org/XML.
24. World Wide Web Consortium (W3C). *HyperText Markup Language (HTML)*. See: www.w3.org/MarkUp.